Automated component testing for agentic AI in medical decision support systems

Waqar Ali[1,4], Benjamin Meyer[1], Linus Stuhlmann[1], Schlomo Aschkenasy[2], Paola Daniore[2], Thilo Stadelmann[1,3], and Ahmed Abdulkadir[1]

[1] ZHAW Centre for Artificial Intelligence, Winterthur, Switzerland
[2] MediRapp AG, Zürich, Switzerland
[3] European Centre for Living Technology, Venice, Italy
[4] Aston University, Birmingham, United Kingdom
{mebr,stmh,stdm,abdk}@zhaw.ch,{w.ali6@aston.ac.uk},
{schlomo,paola@medirapp.ai}

Abstract. Large-language-model-based AI agents that combine retrieval, tool invocation, and generation are often evaluated with aggregate end-to-end scores that provide limited diagnostic information about the source of failures. This is bad practice for agent development because it hinders targeted improvements, prolongs development times, and may let errors slip through. We present a component-wise testing methodology in which system obligations are converted into automated, ground-truth-based tests before open-ended expert evaluation. The protocol separately tests whether required evidence is retrieved, whether the agent selects, orders, and parameterizes tools correctly, and whether the complete workflow solves closed-form integration tasks with exact answers. As a proof of concept, we instantiate the protocol in an agentic cardiology question-answering system that combines guideline retrieval, patient-data access, and computational tools. The resulting component tests show that curated semantic knowledge items improve Hit@5 retrieval by 15–20 percentage points over syntactic chunks, that model choice materially affects tool-use behavior, and that high accuracy on a narrow closed-form task can coexist with weaker tool-use performance across more diverse scenarios. Overall, this work argues that component-wise testing supports development-time evaluation by using automated, verifiable checks to localize failures, guide design choices, and provide practical reliability gates before broader expert evaluation and clinical validation. It should be adopted as a best practice in agent engineering.

Keywords: AI agents · component-wise testing · agentic tool use · medical decision support

1 Introduction

For natural language generation to be useful in communication between health care professionals and patients [1], generated text must be fluent, clinically correct, grounded in patient-specific evidence, and aligned with current medical

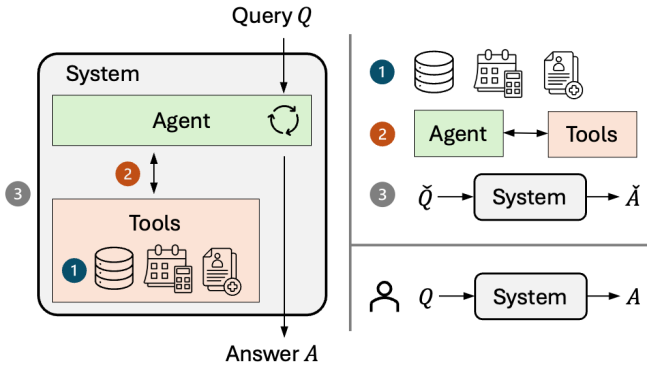


Fig. 1: For the development of a typical tool-assisted agentic system (left), in which an LLM agent interacts with a suite of tools (e.g., databases, calculators) to process a query (Q) and generate an answer (A), we propose automated component testing (top right) before open-ended expert evaluation (bottom right): 1 *Knowledge-retrieval testing:* The retrieval component is tested against predefined evidence ground truth. 2 *Agent-tool-use testing:* The agent's ability to select, order, and parameterize tools is tested against predefined traces. 3 *Closed-form integration testing:* A predefined set of test queries $\check{Q}$ with verifiable answers $\check{A}$ is used to test the full workflow's functional correctness. *Expert evaluation:* Humans rate system responses (A) to real-world queries (Q).

guidelines. In clinical decision support, even small deviations, such as an incorrect threshold, an ignored contraindication, or an outdated recommendation, can invalidate an otherwise plausible response.

Large language models (LLMs) are attractive for clinical question answering and report generation because they produce fluent, context-aware text, but fluency does not guarantee factual accuracy, numerical precision, or guideline adherence [2]. Domain-specialized models can improve medical question answering through fine-tuning and human feedback [3], while tool-assisted agentic systems augment LLMs with retrieval, structured patient-data access, calculators, and explicit reasoning loops [4]. We use *tool-assisted* for systems that combine generation with external tools, and *agentic* for LLM controllers that dynamically select, order, and parameterize these tools. These mechanisms reduce reliance on parametric memory and make intermediate evidence inspectable, but create interacting components whose failures are hard to diagnose from output alone.

End-to-end evaluation can show whether the final answer is correct, but usually not whether an error came from missing guideline evidence, an incorrect tool call, a malformed parameter, or downstream reasoning. LLM-as-a-judge approaches support rapid iteration [5], but their alignment with expert judgment remains uncertain in specialized medical contexts [6]. For tool-assisted clinical systems, evaluation must therefore include diagnostic component-level feedback, not only aggregate scores.

We therefore suggest component-wise testing as a primary development practice for tool-assisted, agentic systems, consistent with earlier work on speech-processing pipelines arguing that unexpected behavior in complex algorithmic chains should be investigated through interpretable intermediate results [2], and with general software engineering and RAG system development practices [7]: Retrieval tests check whether required guideline evidence is surfaced; tool-use tests check whether the agent selects, orders, and parameterizes tools correctly; and closed-form integration tests check whether components interact correctly on tasks with unambiguous answers (see Figure 1). Expert evaluation remains necessary for clinical usefulness and safety, but should follow, rather than replace, these diagnostic tests. As in pipeline error analysis, this decomposition helps identify which component limits performance [8].

In this paper, we seek to answer the following research question: can verifiable component-wise tests provide diagnostic development-time evidence for tool-assisted medical decision-support systems before open-ended expert evaluation We demonstrate the resulting component-first testing protocol on the case study of the development of a tool-assisted agentic cardiology question-answering system. This system combines guideline retrieval, patient-data access, and computational tools for numerical and date-based reasoning. We evaluate two knowledge ingestion strategies, syntactic chunking and curated semantic knowledge items, and compare several open-weight LLMs between $7B$ and $20B$ parameters as tool-orchestrating agents. The case study shows how component tests can guide design choices and reveal weaknesses that a narrow end-to-end score alone may obscure. Based on this case study of verifiable component testing for tool-assisted medical decision support, our contributions are as follows:

(1) *A component-first testing protocol.* We formulate explicit ground-truth checks for retrieval, tool-call trajectories, and closed-form integration testing as automated reliability gates before open-ended expert evaluation.

(2) *An agentic cardiology question-answering case study.* We instantiate this protocol in an agentic cardiology question-answering system that uses an agentic controller with domain-specific retrieval, numerical tools, and patient-data tools.

(3) *An evaluation of knowledge ingestion and model choice* comparing syntactic chunking with curated semantic knowledge items and assessing how different open-weight LLMs affect tool orchestration and closed-form clinical decisions.

(4) *Evidence that component-wise testing reveals hidden tool-use weaknesses* We show that strong performance on a narrow closed-form task can coexist with weaker tool-use behavior across more diverse scenarios, motivating component-wise testing before interpreting end-to-end results.

We conclude that a component-wise testing protocol should be adopted as a best practice in agentic AI engineering.

2 Related work

2.1 Clinical generation and agentic decision support systems

Clinical natural language generation has traditionally relied on rule-based and template-driven systems that convert structured patient data into text, offering strong controllability and precision but requiring extensive manual engineering and offering limited generalization beyond predefined patterns [9]. Recent large language models (LLMs) improve fluency and contextualization, enabling more natural and flexible clinical narratives [10]. However, this flexibility also introduces healthcare-specific risks, including hallucinations, numerical reasoning errors, and unreliable handling of guideline-based constraints [11].

These limitations have motivated a shift toward grounded and interactive decision-support architectures. Retrieval-augmented generation (RAG) has become a dominant paradigm for clinical QA by combining patient-specific context with external medical knowledge, reducing reliance on parametric memory and improving evidence grounding [12]. Recent systems further move beyond single-pass retrieval toward *agentic* architectures, where models iteratively plan, retrieve, reason, and invoke external tools. MedAgent-Pro [12] uses relatively static workflows, whereas CardAIc-Agents [13] implement adaptive multi-turn retrieval and reasoning loops. Similarly, AgentClinic evaluates LLM agents in sequential simulated clinical environments involving patient interaction, multimodal information gathering, and tool use [14].

These architectures introduce interacting retrieval, reasoning, and tool-calling components, making failures harder to localize and motivating component-level evaluation.

2.2 Evaluation of tool-assisted agentic LLM systems

A common approach to automatic evaluation of generative language systems is the use of *LLM-as-a-judge*, where AI models assess generated answers [4]. These methods support scalable comparison and can align with human preferences in general settings [5]. However, their reliability is task- and domain-dependent. Williams et al. [6] showed that LLM-as-a-judge evaluation can diverge from expert judgment in global health contexts, while a RAG study found that end-to-end evaluation still depends heavily on manual assessment [15].

RAG-based tools are highly sensitive to retrieval quality, since missing or incomplete evidence can invalidate otherwise plausible outputs. This has motivated work on knowledge ingestion and chunking strategies, including semantic chunking [16], knowledge-graph-based hierarchies [17], section-aware chunking [18], and structured representations such as tables or decision flows [19]. Established RAG evaluation frameworks also emphasize component-wise assessment: RAGAS separates retrieval- and generation-related dimensions such as context relevance, faithfulness, and answer relevance [7], while RAGChecker provides fine-grained diagnostic metrics for retrieval and generation modules [20]. In addition, the MedRAG-based benchmark evaluates combinations of corpora, re-

trievers and LLM backbones on medical QA datasets, showing that retrieval and corpus design strongly affect downstream performance [21].

Tool-use evaluation is also central for agentic systems. General benchmarks assess whether LLM agents can plan, select tools, and execute API calls correctly [22]. In clinical settings, MedAgentBench evaluates medical LLM agents in a FHIR-compliant virtual EHR environment with physician-authored patient-specific tasks, highlighting the difficulty of reliable tool use in clinical workflows [23]. Clinical calculation is another precision-critical case: Goodell et al. show that task-specific calculation tools substantially reduce LLM errors, supporting the use of machine-readable tools for medical computations [24]. These works demonstrate the importance of component-wise testing, but many frameworks still emphasize broad benchmark performance or LLM-based judgments rather than strictly verifiable tests tied to clinical system obligations.

Despite progress in automated clinical QA, evaluation remains limited for precision-critical domains because end-to-end scores often obscure the source of errors. We adapt and instantiate component-wise, verifiable testing as a development protocol for a cardiology application, with explicit ground-truth checks for retrieval, tool-call trajectories, and closed-form clinical integration.

3 Methods

3.1 Overview of the component-first testing protocol

We use three automated test levels before open-ended expert evaluation: retrieval tests for required guideline evidence, tool-use tests for predefined execution traces, and closed-form integration tests for tasks with exact answers. This decomposition localizes failures to evidence retrieval, tool orchestration, schema handling, or component integration.

Figure 2 summarizes the three automated tests. They are not intended to replace expert evaluation of open-ended clinical responses, but to provide diagnostic reliability gates that can be run repeatedly during development.

3.2 Tool-assisted cardiology question-answering system

We instantiate the protocol in a tool-assisted question-answering system with an agentic controller for cardiovascular decision support. This use case was chosen because guideline-based cardiology questions often require a combination of medical knowledge retrieval, patient-specific measurements, and numerical comparison. Prior work indicates that augmenting language models with retrieval and external tools can improve clinical decision support compared with using a language model alone [25]. Similar to recent medical agentic systems [13], our system uses a dynamic reasoning loop in which the language model can call external tools before producing a final answer.

The system has access to two tool classes. Retrieval tools provide medical guideline and patient-specific information (age, sex, measurements), while computational tools support numerical and date-based calculations. The language model produces structured tool-call instructions that deterministic middleware

parses and executes externally, returning the results as observations for subsequent reasoning.

We evaluate two design choices that are central to this system. The first is the knowledge-ingestion strategy used to construct the retrieval database. The second is the choice of the language model used for tool orchestration and final answer generation. Keeping the overall system architecture fixed, we compare several open-weight models with $7B$–$20B$ parameters that fit within an inference memory footprint of 32 GB. The evaluated models are Ministral 3 Instruct $(14B)$[1] [26], Ministral 3 Reasoning $(14B)$[2] [26], Olmo 3 Instruct $(7B)$[3] [27], Qwen3 $(14B)$[4] [28], and GPT-OSS $(20B)$[5] [29].

3.3 Guideline knowledge ingestion and retrieval

The retrieval database was constructed from a corpus of 20 echocardiographic guidelines. Each knowledge entry was embedded using the open Nomic embedding model[6] [30], producing 768-dimensional embeddings. At inference time, the incoming query was embedded with the same model and used as the search vector for nearest-neighbor retrieval from the vector database. We intentionally used a simple retrieval pipeline without re-ranking or query refinement so that the effect of the knowledge-ingestion strategy could be assessed directly.

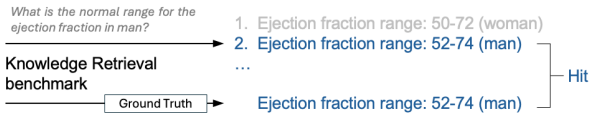
We compared two ingestion strategies. The first segmented guideline documents into fixed token windows, yielding syntactic chunks. This common RAG baseline can separate table entries, thresholds, and explanatory context that belong together clinically. The second converted the guidelines into curated semantic knowledge items via an AI-assisted process using ChatGPT (GPT-5.1). The process comprised three prompting stages: identifying the major diagnostic steps in the guidelines, listing diagnostic aspects within each step, and summarizing the key diagnostic thresholds for each aspect. This yielded 95 topical knowledge items with compact descriptions of clinically relevant diagnostic values.

These items were designed for testing and system development rather than clinical deployment, and were not exhaustively clinician-reviewed. However, the specific diagnostic thresholds used in the tests were checked against the source guideline material before being used as ground truth.

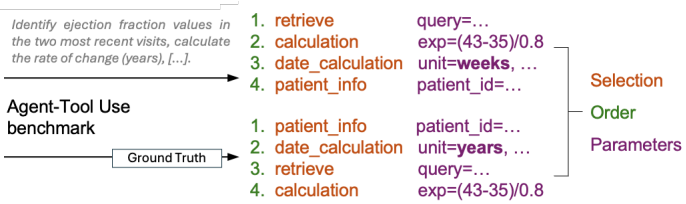
3.4 Knowledge-retrieval testing

Knowledge retrieval grounds the LLM agent's reasoning in established medical guidelines by bringing relevant evidence into context before generation. Testing this component in isolation is essential because an end-to-end answer can fail either because the model did not receive the required evidence or because it failed to use available evidence correctly.

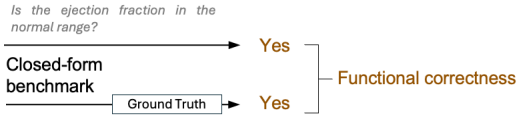
[1] https://huggingface.co/mistralai/Ministral-3-14B-Instruct-2512
[2] https://huggingface.co/mistralai/Ministral-3-14B-Reasoning-2512
[3] https://huggingface.co/allenai/Olmo-3-7B-Instruct
[4] https://huggingface.co/Qwen/Qwen3-14B
[5] https://huggingface.co/openai/gpt-oss-20b
[6] https://www.nomic.ai/news/nomic-embed-text-v1



(a) *Knowledge retrieval.* A benchmark query with predefined ground truth is submitted to the retrieval component. The test checks whether the relevant guideline item appears among the retrieved items.



(b) *Agent-tool use.* A benchmark query with a predefined execution trace is submitted to the agent. The predicted tool calls are compared with this trace for tool selection, execution order, and parameters.



(c) *Closed-form integration.* A verifiable clinical scenario is submitted to the complete system. The generated structured answer is matched against the ground-truth label.

Fig. 2: Illustration of the three automated tests used in this study.

We constructed a benchmark of 50 clinical queries: 30 simple queries targeting single diagnostic concepts (e.g., normal ejection-fraction reference values) and 20 complex requiring multiple related measurements or interpretation rules. The relevant guideline item(s) were manually specified as ground truth for each query. Figure 2a illustrates that the retrieval component was queried independently of the full system and its results compared against this ground truth. Retrieval performance was quantified using Hit@k. A query was counted as successful if the manually specified ground-truth item appeared within the top-k retrieved items.

3.5 Agent-tool-use testing

Tool use refers to the model emitting structured tool-call instructions, such as JSON-like arguments or source-code-like commands, which deterministic middleware parses, executes externally, and returns as observations to the model.

We curated a benchmark of 100 clinical scenarios, split into 50 simple and 50 complex tasks. Each scenario pairs a clinical query with a ground-truth execution trace specifying the required tool sequence and expected parameters. As shown in Figure 2b, the agent's predicted trajectory is compared with this trace.

We test three dimensions of tool-use correctness. *Selection* measures whether the agent identifies the required tool(s); e.g., a query about normal ejection-fraction ranges should call the guideline retrieval tool rather than the patient-data tool. *Order* measures whether tools are invoked in a dependency-consistent sequence; e.g., computing a trend over prior ejection-fraction measurements requires retrieving them before calling a numerical calculator. *Parameters* correctness measures whether the generated arguments match the expected schema and values; e.g., a calculator call for adding 75 and 25 should pass `"expression": "75 + 25"`, with malformed or incorrect values counted as errors.

The middleware includes an automatic retry mechanism with at most three attempts to recover from basic syntax or parsing errors, and we treat successful execution within this limit as part of agent behavior. To quantify reliance on retries, we report the mean reciprocal rank of the first successful attempt, computed only over cases that were eventually completed successfully. For each successful case, the score is calculated as $(1/r)$, where (r) is the attempt number at which the first success occurs. A value of 1.0 therefore indicates success on the first attempt, while lower values indicate that additional attempts were required before successful execution. We also record failure cases where no valid command is executed.

3.6 Closed-form integration testing

After testing retrieval and tool use separately, we test the complete workflow on a closed-form task with exact answers. Closed-form integration testing does not capture the full range of open-ended clinical interactions, but it provides an objective integration test for the agentic workflow. It checks whether the system can combine patient-data retrieval, guideline retrieval, numerical comparison, and constrained answer generation in a single task.

The closed-form benchmark consists of 1000 synthetic patient cases. For each case, the system must determine whether the patient's left ventricular ejection fraction (LVEF) is within the normal range. A correct response requires the following sequence of operations: (i) retrieve the patient's LVEF value and biological sex using the patient-data tool, (ii) retrieve the applicable normal ranges using the guideline-retrieval tool, (iii) compare the patient value with the retrieved threshold, and (iv) output a constrained binary answer, *Yes* for normal or *No* for abnormal. The generated answer is matched exactly against the ground-truth label, as illustrated in Figure 2c.

We report accuracy, F1 score, precision, and recall for this closed-form integration test. These metrics are interpreted as evidence of functional correctness on the constrained LVEF task, not as evidence of general clinical validity.

For binary proportions, we report Wilson score 95% confidence intervals [31] where appropriate. The intervals are computed from the number of successes and evaluated cases for Hit@k, tool-selection accuracy, and closed-form accuracy. The intervals are used descriptively to indicate finite-benchmark uncertainty.

4 Results

The automated tests provide diagnostic evidence for two design choices in the tool-assisted system: the knowledge-ingestion strategy and the language model used for tool orchestration and closed-form answering. Table 1 reports retrieval performance for syntactic chunks and curated knowledge items. Curated items improve Hit@5 by 15 to 20 percentage points for both simple and complex queries, indicating that restructuring guideline text into compact, measurement-centered units improves retrieval of relevant evidence. Tables 2 and 3 summarize the effect of model selection. Table 2 reports each model's ability to use the custom tools for simple and complex clinical queries.

Table 1: Knowledge-retrieval test results for two knowledge ingestion methods with the Nomic embedding model on 30 simple and 20 complex queries. Wilson 95% confidence intervals are shown in brackets.

Chunking	**Query**	**Hit@1**	**Hit@3**	**Hit@5**
Syntactic chunks	Simple	0.30 [0.17, 0.48]	0.63 [0.46, 0.78]	0.77 [0.59, 0.88]
	Complex	0.20 [0.08, 0.42]	0.60 [0.39, 0.78]	0.70 [0.48, 0.85]
Knowledge item	Simple	0.57 [0.39, 0.73]	0.80 [0.63, 0.90]	0.97 [0.83, 0.99]
	Complex	0.50 [0.30, 0.70]	0.70 [0.48, 0.85]	0.85 [0.64, 0.95]

Table 2: Agent-tool-use test results on the tool-use benchmark (100 queries, equally split between simple and complex clinical tasks). We report accuracy as proportions for tool selection, calling order, parameter correctness, MRR, and the number of overall failures across open-weight LLMs.

Model	**Select.**	**Order**	**Param.**	**MRR**	**Fail.**
Simple queries					
Ministral 3 Instruct (14B)	0.98 [0.90, 1.00]	0.96	1.00	0.99	0
Ministral 3 Reasoning (14B)	0.94 [0.84, 0.98]	0.92	1.00	1.00	0
Olmo 3 Instruct (7B)	0.58 [0.44, 0.71]	0.56	0.88	1.00	15
Qwen3 (14B)	0.98 [0.90, 1.00]	0.96	1.00	0.99	0
GPT-OSS (20B)	1.00 [0.93, 1.00]	1.00	1.00	1.00	0
Complex queries					
Ministral 3 Instruct (14B)	0.917 [0.81, 0.97]	0.854	1.00	0.99	1
Ministral 3 Reasoning (14B)	0.913 [0.81, 0.97]	0.891	1.00	1.00	4
Olmo 3 Instruct (7B)	0.729 [0.58, 0.83]	0.479	0.72	0.76	4
Qwen3 (14B)	0.920 [0.81, 0.97]	0.900	1.00	1.00	2
GPT-OSS (20B)	0.960 [0.87, 0.99]	0.920	1.00	1.00	0

Most evaluated models achieve about 90% accuracy or higher on tool selection, order, and parameterization. GPT-OSS (20B) performs best, with no failures; Ministral 3 Instruct, Ministral 3 Reasoning, and Qwen3 are also strong. Olmo 3 Instruct (7B), the smallest model, performs worse, especially on tool selection and ordering.

Table 3: Accuracy, F1, precision, and recall of the closed-form integration test across the considered LLMs.

Model	Accuracy	F1	Precision	Recall
Ministral 3 Instruct ($14B$)	0.98 [0.97, 0.99]	0.99	0.98	0.99
Ministral 3 Reasoning ($14B$)	0.93 [0.91, 0.94]	0.96	0.93	0.99
Olmo 3 Instruct ($7B$)	0.96 [0.95, 0.97]	0.97	1.00	0.95
GPT-OSS ($20B$)	0.99 [0.98, 0.99]	0.99	1.00	0.99

These results also reveal behavior not visible from the closed-form task alone. Olmo 3 Instruct shows limited tool-selection accuracy on the broader benchmark (58–73%) yet reaches 96% on the closed-form test, indicating that success on a single well-defined task does not imply robust tool use across diverse, multi-step scenarios. It also performs 15 percentage points better on complex queries than simple ones, echoing the observation that success on harder generative-AI tasks does not guarantee success on easier ones [32]. While the mechanism is unclear, this further motivates use-case-specific component testing.

Table 3 presents the closed-form integration test on the verifiable LVEF scenario, reporting accuracy, F1, precision, and recall across 1000 synthetic patient verdicts. All models exceed 90 percent accuracy and F1, with GPT-OSS ($20B$) achieving the highest accuracy, showing that the evaluated models can perform well on this constrained task when given the required patient data, guideline evidence, and tool interfaces.

5 Discussion and conclusions

This case study shows that component-wise tests make failures in tool-assisted medical decision support more localizable than aggregate scores. Curated knowledge items improved retrieval over syntactic chunks, whereas model choice substantially affected tool selection, ordering, and parameterization. The Olmo 3 Instruct result illustrates why narrow closed-form integration tests can overestimate readiness: high LVEF-task accuracy coexisted with weaker tool-use behavior across diverse scenarios.

The proposed tests should be interpreted as development gates rather than replacements for clinical validation: Retrieval tests check whether the system surfaces required guideline evidence, tool-use tests check whether selection, ordering, and parameterization are correct, and closed-form integration tests check whether these components combine correctly in constrained, verifiable scenarios. Passing does not establish clinical safety, but failing identifies concrete weaknesses to address before broader expert evaluation.

This workflow is particularly valuable for agentic systems where failures can stem from missing evidence, inappropriate retrieval, incorrect tool calls, or erroneous reasoning. Isolating these obligations supports faster iteration and more transparent model selection.

Several limitations apply. The evaluation covers one cardiology use case and a narrow synthetic LVEF classification task; it does not assess open-ended clinical questions, contraindications, treatment recommendations, or uncertainty-communication. The retrieval and tool-use benchmarks are also limited in size, so reported percentages should be interpreted cautiously.

Wilson intervals quantify this uncertainty: they are wider for smaller retrieval and tool-use subsets and narrower for the 1000-case closed-form test. We use them to assess design signals, not to overinterpret small differences among closely performing models.

The study evaluates automated component performance, not deployment readiness, and does not replace clinician review, prospective evaluation, or safety analysis in real clinical workflows. Many of the limitations can be addressed in future work by expanding the benchmark set, including additional clinical tasks, reporting uncertainty estimates, analyzing failure modes, and conducting additional expert evaluations.

Automated component testing provides a practical way to assess tool-assisted agentic LLM systems before costly expert evaluation. In this cardiology case study, separate retrieval, tool-use, and closed-form integration tests exposed design differences that would be difficult to localize from an aggregate end-to-end score alone. These findings support a component-first testing workflow in which automated, verifiable tests are used as prerequisites for broader expert evaluation and clinical validation. We conclude that a suitable component-wise testing protocol should be adopted as a best practice in the engineering of any LLM-based agentic AI system.

References

[1] A. J. Cawsey et al., "Natural Language Generation in Health Care," *Journal of the American Medical Informatics Association*, vol. 4, no. 6, pp. 473–482, 1997.

[2] T. Stadelmann, "A guide to AI. Understanding the technology, applying it successfully, and shaping a positive future," Jan. 2025.

[3] K. Singhal et al., "Large language models encode clinical knowledge," en, *Nature*, vol. 620, no. 7972, pp. 172–180, Aug. 2023.

[4] W. Wang et al., *A Survey of LLM-based Agents in Medicine: How far are we from Baymax?* arXiv:2502.11211, May 2025.

[5] L. Zheng et al., "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena," *NeurIPS*, vol. 36, pp. 595–623, Dec. 2023.

[6] G. Williams et al., *Human Evaluators vs. LLM-as-a-Judge: Toward Scalable, Real-Time Evaluation of GenAI in Global Health*, en, Oct. 2025.

[7] S. Es et al., "Ragas: Automated evaluation of retrieval augmented generation," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, 2024, pp. 150–158.

[8] L. Baresi et al., "An Introduction to Software Testing," *Electronic Notes in Theoretical Computer Science*, FoVMT, vol. 148, no. 1, pp. 89–111, Feb. 2006.

[9] G. T. Berge et al., "Combining unsupervised, supervised and rule-based learning: The case of detecting patient allergies in electronic health records," *BMC Medical Informatics and Decision Making*, vol. 23, no. 1, p. 188, Sep. 2023.

[10] M. Lyu et al., *Natural Language Generation in Healthcare: A Review of Methods and Applications*, arXiv:2505.04073 [cs], May 2025.
[11] Z. A. Nazi et al., *Large language models in healthcare and medical domain: A review*, en, arXiv:2401.06775 [cs], Jul. 2024.
[12] Z. Wang et al., *MedAgent-Pro: Towards Evidence-based Multi-modal Medical Diagnosis via Reasoning Agentic Workflow*, arXiv:2503.18968 [cs], Jul. 2025.
[13] Y. Zhang et al., *CardAIc-Agents: A Multimodal Framework with Hierarchical Adaptation for Cardiac Care Support*, arXiv:2508.13256 [cs], Aug. 2025.
[14] S. Schmidgall et al., "AgentClinic: A multimodal agent benchmark to evaluate AI in simulated clinical environments," 2024.
[15] L. Brehme et al., "Retrieval-Augmented Generation in Industry: An Interview Study on Use Cases, Requirements, Challenges, and Evaluation," *arXiv preprint arXiv:2508.14066*, 2025.
[16] R. Qu et al., "Is semantic chunking worth the computational cost?" In *Findings of the Association for Computational Linguistics*, NAACL, 2025, pp. 2155–2177.
[17] J. Wu et al., "Medical Graph RAG: Evidence-based Medical Large Language Model via Graph Retrieval-Augmented Generation," in *Annual Meeting of the Association for Computational Linguistics*, Jul. 2025, pp. 28 443–28 467.
[18] T. Chen et al., "Dense x retrieval: What retrieval granularity should we use?" In *Empirical Methods in Natural Language Processing*, 2024, pp. 15 159–15 177.
[19] C. A. Gomez-Cabello et al., "Comparative Evaluation of Advanced Chunking for Retrieval-Augmented Generation in Large Language Models for Clinical Decision Support," en, *Bioengineering*, vol. 12, no. 11, p. 1194, Nov. 2025.
[20] D. Ru et al., "RAGChecker: A Fine-grained Framework for Diagnosing Retrieval-Augmented Generation," *Advances in Neural Information Processing Systems*, vol. 37, pp. 21 999–22 027, Dec. 2024.
[21] G. Xiong et al., "Benchmarking Retrieval-Augmented Generation for Medicine," in *Findings of the Association for Computational Linguistics ACL 2024*, Association for Computational Linguistics, 2024, pp. 6233–6251.
[22] W. Li et al., *Adaptive Tool Use in Large Language Models with Meta-Cognition Trigger*, Version Number: 2, 2025.
[23] Y. Jiang et al., *MedAgentBench: A Realistic Virtual EHR Environment to Benchmark Medical LLM Agents*, Feb. 2025.
[24] A. J. Goodell et al., "Large language model agents can use tools to perform clinical calculations," *npj Digital Medicine*, vol. 8, no. 1, p. 163, Mar. 2025.
[25] D. Ferber et al., "Development and validation of an autonomous artificial intelligence agent for clinical decision-making in oncology," *Nature Cancer*, vol. 6, no. 8, pp. 1337–1349, Aug. 2025.
[26] A. H. Liu et al., *Ministral 3*, arXiv:2601.08584 [cs], Jan. 2026.
[27] T. Olmo et al., *Olmo 3*, arXiv:2512.13961 [cs], Dec. 2025.
[28] A. Yang et al., *Qwen3 Technical Report*, arXiv:2505.09388 [cs], May 2025.
[29] OpenAI et al., *Gpt-oss-120b & gpt-oss-20b Model Card*, arXiv:2508.10925 [cs], Aug. 2025.
[30] Z. Nussbaum et al., "Nomic Embed: Training a Reproducible Long Context Text Embedder," en, *Transactions on Machine Learning Research*, Nov. 2024.
[31] E. B. Wilson, "Probable Inference, the Law of Succession, and Statistical Inference," *Journal of the American Statistical Association*, vol. 22, no. 158, pp. 209–212, 1927.
[32] Z. Yang et al., "Can large language models always solve easy problems if they can solve harder ones?" *arXiv preprint arXiv:2406.12809*, 2024.